Named Data Networking: File Interception in an NDN environment

Purva Bandekar, Sneha Regmi

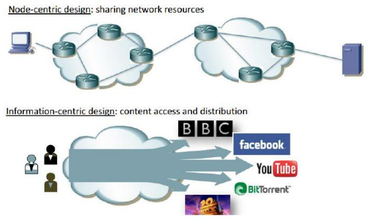
Department of Electrical Engineering, San Jose State university, San Jose, California 95192.

Abstract

Named Data Networking (NDN), a subset of Information Centric Networking (ICN), is a network architecture that revamps network service semantics from “deliver packet to given destination” to “retrieve data with given name.” This project aims at implementing an NDN test-bed in a Software Defined Networking (SDN) context using OpenFlow and POX controller. Key aspects of NDN are implemented in a uniquely SDN manner. For example an "interest" sent from the client to get particular information is intercepted by an OpenFlow enabled switch which forwards this "request" to the SDN controller for processing by our NDN SDN application, after which the real information repository location/URL is sent back to the client. Subsequently, the client accesses the information source to fetch the data. This shift to an SDN approach to implementing NDN as compared to extensive modifications to routers and routing protocols such as OSPF can greatly accelerate the availability of the information centric networking paradigm.

Introduction

The present day internet architecture was built around the fundamental objective of being able to present a communication model that was solely host-centric. This seemed to work fine considering the early day internet users. Today, however we have shifted from a host centric approach to a much more information-centric model. With the constant evolution of the Internet, people are more concerned about accessing various kinds of information (be it texts, pictures, videos, etc.) irrespective of its physical location [1]. This clearly brings us to realize the need of a newer Internet architecture that focuses more on getting information to the users rather than looking at the physical location of where this information could be residing in. This need has led to the development of NDN, which is an emerging architecture that has proved to be dynamic, cost effective and manageable [1].

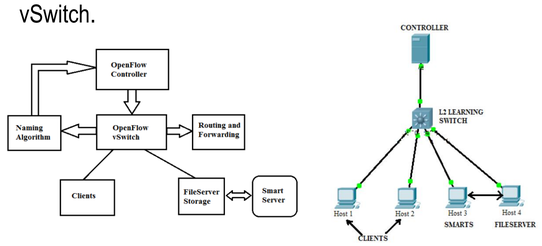


NDN lies on the concepts of Content Centric Networking (CCN). In NDN, the prime focus is getting the packet with the content to the client [2]. Therefore, the NDN architecture is largely based on two important packets [2] :

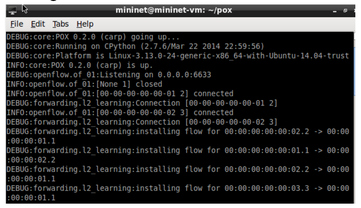
- The “interest packet” that carries the interest to fetch
- The “data packer” that carries the data

General Solution Approach

We have tested on a network topology consisting of four hosts, an Open Flow vSwitch and a POX controller using Mininet. The diagrams below portray our approach in implementing NDN topology using POX as the controller and local information server connected via an OpenFlow vSwitch.

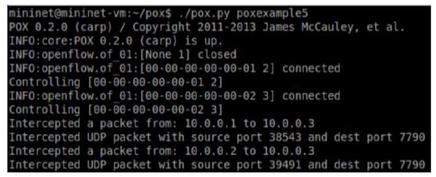


In the POX controller topology as the first packet arrives at the switch it is diverted to the controller, where the controller maintains a hash table that maps the address to the output port. In our topology the hash table contains both the output port and the MAC address. As soon as the switch receives the first packet from the host, it updates the address in the port table. In case, the address is a multicast address the controller makes a decision to flood it on all output ports. In addition if there is no table entry for the destination for that packet then it again broadcasts the message on all output ports. If both the source and destination addresses are the same then the controller instructs the switch to drop the packet. Otherwise, the controller installs the flow table entry corresponding to that destination address and output port. Installing that flow table entry in that switch prevents future packets to that flow from being redirected to the controller and the switch can now handle packets to that destination for succeeding requests. The figure below shows the controller up and running and installing flows.



The controller installs flows as soon as the switch comes up and connects to the controller. Only when this is done, the host h1 (or h2) is able to send the “Interest” packet to the Smart Server h3 that is listening on a UDP port. Consecutively, the “Interest” packet is intercepted by the POX controller and it gives information about the source and destination of the packet. The Interest message is basically a UDP packet what is sent form h1 (or h2) to the Smart Server.

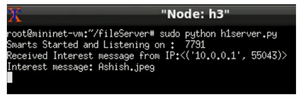
The Open Flow switch then intercepts this packet and sends it to the POX controller. The controller then notes the source and destination ip addresses and the respective ports. The figure below shows this interception.

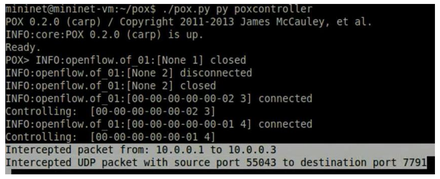


After the packet is intercepted by the controller, we re-created the UDP packet and sent the packet with the same data payload to the “Smart Server”. Following this, the Smart server h3 tries to see if there is a match in the dictionary and tries to fetch the required file of interest. To be able to create a file “fetching scenario, we decide to use the HTTP protocol and make the information server i.e. h4 as a simple HTTP server.

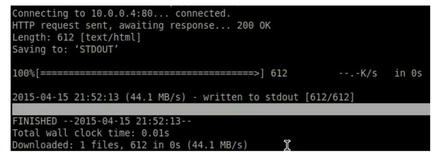
Results

In our topology, h4 acts as the information server and h3 acts as the Smart server together with h1 and h2 behaving as hosts that request for the information. A very important aspect of the fetching process is the use of a dictionary in the Smart Server. How we have implemented the file fetching process is that once h1/h2 requests for a particular interest, then the Smart Server will check for that interest in the dictionary and incase, there is a match then, it will send the information server’s address (information server where the file of interest is stored). In case, there is no match then, it will inform h1/h2 that the file of interest is not present in the information server. Now, what makes our program very dynamic is that we have taken into consideration information registering. What this means is that, every time a new information i.e. a new file is stored in the information repository h4, that information will be updated in the dictionary as well thus providing a dynamic update of the latest information in the server. Below is an example when the host h1 requests for the interest “Ashish.jpg”

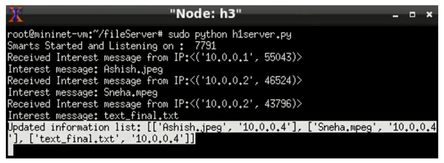




The figure below shows the file of interest being successfully fetched from the information server repository.



Moreover, we have also taken information registering into consideration. Whenever a new information is added in the information server h4, the dictionary in the smart server h3 gets automatically updated as is shown in the figure below.



Conclusion

NDN exhibits an application-driven development such that the verification and validation of performance and functional advantages of NDN, such as routing on names promotes efficient authoring of complex distributed applications thus reducing errors, time and expense of design and deployment. We implemented a prototypical NDN-based network topology that demonstrates OpenFlow routing and forwarding strategies and content fetching with the help of namespaces. Our project has so far served us in learning detail about the key concepts of Named Data Networking and we have demonstrated setting up the POX controller, installing flows in OpenFlow switch and debugging the flows, creating Hash maps/dictionaries for referring to namespaces, dynamically expanding content discovering hash maps for new content and so on.

Key References

[1] Professor George Pavlou, University College London, “Information-Centric Networking: Overview, Current State and Key Challenges”, COMET-Envision Workshop Keynote.

[2] Lixia Zhang, Alexander Afanasyev, Jeffrey Burke, Van Jacobson, Kc Claffy, Lan Wang, Patrick Crowley, Beichuan Zhang, "Named Data Networking",vol 44. no.3, July 2014.

Acknowledgements

We would like to thank our project advisor, Professor Greg Bernstein for all the help and guidance that he provided us throughout this project. Also, we are very thankful to the other team who worked with us on this project: Ashish Chakrabortty and Karthik Kumar for all the help and the great teamwork during the entire project period.